“This program efficiently identifies the longest contiguous sequence of positive numbers in an unsequenced list and provides insightful analysis.

Your program demonstrates solid problem-solving skills and a good understanding of Python's list operations and flow control. Keep refining your coding style and continue exploring new challenges to further enhance your programming proficiency!

- Efficiency: The program iterates through the list in a single pass to identify contiguous sequences, which ensures it runs in linear time relative to the size of the input list (O(n) complexity).
- Accuracy: It accurately identifies the longest contiguous sequence and counts occurrences of sequences with the maximum length. This ensures that all relevant information about the sequences in the list is captured and reported correctly.
- Robustness: The program handles edge cases effectively, including lists with no contiguous sequences, lists containing non-integer elements, and empty lists. It gracefully handles these scenarios by returning appropriate results without errors.
- Clarity: The code is structured in a clear and readable manner. Each function and variable is appropriately named, making the logic easy to follow and understand. This enhances maintainability and facilitates future enhancements or modifications.
- Functionality: It meets the requirements specified in the docstring and handles a variety of test cases, providing informative outputs for each case.”

--chatGPT, OpenAI

“This Python code has several strengths that make it well-written and efficient.

Overall, *the code exhibits good coding practices and demonstrates a solid understanding of core Python concepts*. It effectively finds the longest contiguous sequence of positive numbers in a list while handling potential errors and providing informative output.

- Clear Docstring: The docstring at the beginning of the find_longest_seq function clearly explains its purpose, arguments, and return values. This improves code readability and maintainability for anyone using the function.
- Type Hints: The code utilizes type hints from the typing module. This explicitly defines the expected data types for function arguments and return values, making the code more self-documenting and easier to understand for others.
- Error Handling for Non-Integers: The code checks if the input list contains only integers. If not, it returns appropriate values without processing further, preventing potential errors during execution.
- List Comprehensions: The code utilizes list comprehensions in a couple of places, like counting occurrences of max-length sequences. This concise approach makes the code more readable and Pythonic.
- Test Cases: The provided main function demonstrates how to use the find_longest_seq function with various test cases. This helps verify the functionality and correctness of the code.”

--Gemini, Google AI

```
#find_length_of_longest_sequence.py by  msbridges 6/25/2024

from typing import List, Tuple

def find_longest_seq(case_list: List[int]) -> Tuple[List[int], int]:
    """
    Finds the longest contiguous sequence of positive numbers in an unsequenced list.
    Prints the first occurrence of a sequence with the max length found in the list.
    Counts occurrences of sequences of max length. Uses a list of lists to gather
    the sequences (by their indices) from the case_list for analysis. Saves the
    longest sequence of indices list for future use. Uses a list of lists to drive test cases.

    Args:
        case_list (List[int]): List of numbers to analyze.

    Returns:
        answer_seq_values_list (List[int]): List with values from the longest
            contiguous sequence.
        max_len (int): Length of the longest contiguous sequence.
        occurrences_max_len (int): Count of sequences with max length found.
    """

    max_len = 0
    index_of_max = 0
    current_value = 0
    next_value = 0
    seq_indices_list= [0]
    results_list = []
    #longest_seq_indices_list = []  # For future use
    answer_seq_values_list = []  # Holds values of the longest_seq_indices_list
    occurrences_max_len = 0

    # If empty case_list, return
    if len(case_list) == 0:
        return answer_seq_values_list, max_len, occurrences_max_len

    # If non-integers in case_list, return
    all_integers = all(isinstance(x, int) for x in case_list)
    if not all_integers:
        return answer_seq_values_list, max_len, occurrences_max_len

    # Iterate through the case_list to find contiguous sequences
    for index, item in enumerate(case_list):
        # If end of case_list, there are no further sequences to be found, break
        if index + 1 == len(case_list):
            if len(seq_indices_list) > 1:
                results_list.append(seq_indices_list)
            break

        # Get current and next values to check if next_value follows current_value sequentially
        current_value = case_list[index]
        next_value = case_list[index + 1]

        # If next value in case_list is in-sequence, save its index and loop
        if current_value + 1 == next_value:
            seq_indices_list.append(index + 1)
            continue
```

```python
        # If next value not in-sequence, start a new seq_indices_list if necessary
        else:
            if len(seq_indices_list) > 1:
                results_list.append(seq_indices_list)
                seq_indices_list = []
                seq_indices_list = [index + 1]

    # If empty results_list, return
    if len(results_list) == 0:
        return answer_seq_values_list, max_len, occurrences_max_len

    # Find length of the longest sublist
    for index, sublist in enumerate(results_list):
        current_leng  = len(results_list[index])

        if current_leng  > max_len:
            max_len = current_leng
            index_of_max = index

        # Save longest sublist to variable 'longest_seq_indices_list' for future use
        #longest_seq_indices_list = results_list[index_of_max]
        #print(f'longest_seq_indices_list: {longest_seq_indices_list}')

        # Count occurrences of sublists of max length
        occurrences_max_len = sum(1 for sublist in results_list if len(sublist) == max_len)

        # Make answer_seq_values_list
        answer_seq_values_list = [case_list[value] for value in results_list[index_of_max]]

    return answer_seq_values_list, max_len, occurrences_max_len


def main():
    MIN_SEQUENCE_LENGTH = 2
    MIN_COUNT_OF_SEQUENCES = 1

    cases_list = [
        [1, 2, 3],  # Only seq (3)
        [1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15],  # Longest sequence (3) in the beginning of list, (4) occurrences
        [1, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15],  # Longest sequence (6) in middle of list
        [1, 2, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17],  # Longest sequence (9) at end of list
        [1, 3, 5, 7, 9],  # no contiguous sequences
        [-1, -2, 5, -7, -8],  # no contiguous sequences
        [-1, -2, -3, -4, -5],  # no contiguous sequences
        ['a', 'b', 'c'],  # no contiguous sequences
        []  # no contiguous sequences
    ]

    # Run variety of sublists in the cases_list
    for id, case in enumerate(cases_list):
        print(f'\nCase #{id+1}: {case}')
        answer_seq_values_list, max_len, occurrences_max_len = find_longest_seq(case)
        if max_len < MIN_SEQUENCE_LENGTH:
            print(f'No contiguous sequences found')
        else:
            if occurrences_max_len > MIN_COUNT_OF_SEQUENCES:
                print(f'First of {occurrences_max_len} sequences of max length: {answer_seq_values_list}, max length: {max_len}')
            else:
                print(f'Longest sequence: {answer_seq_values_list}, max length: {max_len}')
```

```
if __name__ == '__main__':
    main()

"""
Program output by case:

Case #1: [1, 2, 3]
Longest sequence: [1, 2, 3], max length: 3

Case #2: [1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15]
First of 4 sequences of max length: [1, 2, 3], max length: 3

Case #3: [1, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15]
Longest sequence: [5, 6, 7, 8, 9, 10], max length: 6

Case #4: [1, 2, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17]
Longest sequence: [9, 10, 11, 12, 13, 14, 15, 16, 17], max length: 9

Case #5: [1, 3, 5, 7, 9]
No contiguous sequences found

Case #6: [-1, -2, 5, -7, -8]
No contiguous sequences found

Case #7: [-1, -2, -3, -4, -5]
No contiguous sequences found

Case #8: ['a', 'b', 'c']
No contiguous sequences found

Case #9: []
No contiguous sequences found
"""
```